POLITECNICO
MILANO 1863

# Dreamy Tales

Design Document
DIMA 2023/24

Gabriele Shu (11016752), Jaskaran Ram (10735884)

Feb 4, 2023

# Index

POLITECNICO
MILANO 1863

# 1 Introduction

## 1.1 Document Purpose

The main purpose of the present document is to support the development team in the realization of the system to be. It provides an overall description of the adopted system architecture and a breakdown of the various components down to a significant extent, also describing their interactions with each other. On top of that, the Design Document (DD) describes the implementation, integration and testing plans which are defined keeping into account priority, required effort, and impact of the single components.

## 1.2 Project Scope

The "Dreamy Tales" application represents a visionary venture into the realm of children's storytelling, integrating cutting-edge artificial intelligence (AI) technology to craft an immersive and educational experience. At its core, "Dreamy Tales" is designed to captivate the imagination of young audiences while fostering learning and personal development. Here's a comprehensive overview of its scope:

The App is dedicated to delivering an extensive collection of captivating stories tailored to the diverse interests and developmental stages of children. Ranging across various genres and themes, these stories serve not only as sources of entertainment but also as avenues for learning and exploration.

Central to the application's innovation is its utilization of AI algorithms which offers extensive customization options, empowering users to tailor the storytelling experience to their preferences and developmental needs. From selecting story themes and character traits to choosing language preferences, users can personalize their journey through the narrative landscape.

Dreamy Tales is a cross-platform application crafted for both iOS and Android platforms, ensures seamless access and convenience for its users, enabling them to embark on storytelling adventures anytime, anywhere.

## 1.3 Platform: Flutter and Firebase

We have strategically chosen Flutter and Firebase as the foundational technologies for the Dreamy Tale app, ensuring a seamless and efficient development process.

**1. Flutter for Cross-Platform Excellence:**

Flutter, with its single codebase for both iOS and Android platforms, aligns perfectly with our goal of delivering a consistent and high-quality user experience across devices. The framework's hot-reload feature expedites development, allowing for rapid iterations and efficient debugging. Flutter's expressive UI and native performance contribute to the creation of an elegant and responsive application.

**2. Firebase for Scalability and Real-Time Capabilities:**

Firebase serves as our backend solution, providing a scalable and reliable infrastructure for Dreamy Tale. Its real-time database ensures instant updates, enabling collaborative storytelling experiences. Firebase Authentication enhances user security, allowing for seamless and secure login processes. Additionally, Firebase Cloud Functions support serverless architecture, optimizing scalability and responsiveness.

## 1.4 Definitions, Acronyms, Abbreviations

### 1.4.1 Definitions

- **User**: an account registered in the application with email and password.
- **Logged User**: fixed the application device, a user that is currently logged inside the application using that device.
- **Child profile**: is a set of information about a child, including age, gender, avatar, and name, used to customize and tailor an experience within an application or system.

- **Character**: is an individual within the narrative. Each character has specific details, including age, name, gender, and role.
- **Story**: is a personalized bedtime story

### 1.4.2 Acronyms

• DD: Design Document
• JWT: JSON Web Tokens
• API: Application Programming Interface
• REST: Representational state transfer
• DTO: Data Transfer Object
• e-Mall: e-Mobility for All.
• UI: User interface.

### 1.4.3 Abbreviations

- BIO: Biography
- APP: Application
- AI: Artificial Intelligence

## 1.5 Revision history

- Version 1.0 (February 4th, 2024: first version);

## 1.6 Reference Documents

- Slides of Software Engineering 2 course on WeBeep.
- DD Sample from A.Y. 2020-2021
- Slides of Desing and Mobile Application course held by Professor Luciano Baresi.

## 1.7 Document Structure

Mainly the current document is divided in 5 chapters, which are:

1. **Introduction**: The first chapter includes the introduction in which is explained the purpose of the document. Finally, important information for the reader is given, i.e. definitions, acronyms, synonyms and the set of documents referenced.

2. **Features and Requirements**

3. **Architectural Design**: it includes a detailed description of the architecture of the system, including the high-level view of the elements, the software components, a description through runtime diagrams of various functionalities of the system and, finally, an in-depth explanation of the architectural pattern used.

   **User Interface Design**: are provided the mockups of the application user interfaces, with the links between them to help in understanding the flow between them.

4. **Application Services**: Cloud Storage, external API, local preferences, interactions between components.

5. **Implementation, Integration and Test Plan**: it describes the process of implementation, integration and testing to which developers have to stick in order to produce the correct system in a correct way.

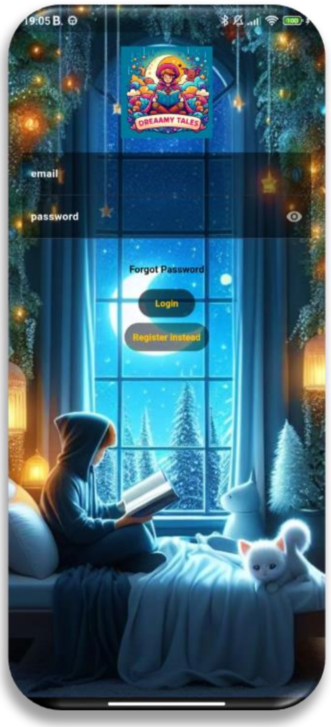
6. **Effort spent**

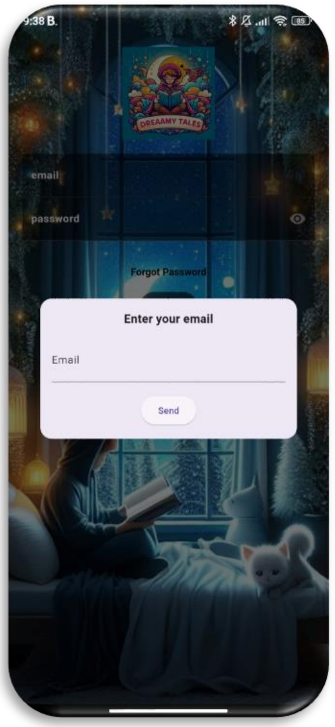
7. **References**: it contains the references to any documents and to the Software used in this document.

# 2 Features and requirements

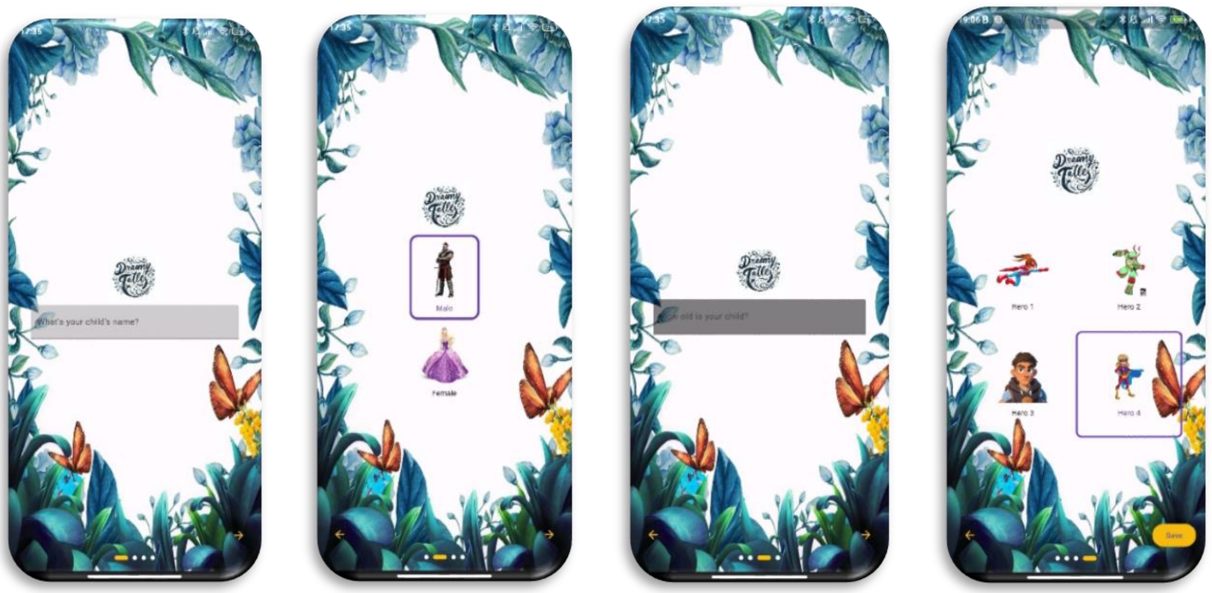
## 2.1 Features Overview

1. **Authentication Process through Firebase:**
   Dreamy Tale ensures a secure and user-friendly authentication process leveraging Firebase. Users can seamlessly navigate through login, register, and forget password functionalities, guaranteeing a safe and personalized experience.





**2. Child Profile Creation:**
   Parents can create personalized profiles for their children, including name, age (0-10), gender, and avatar selection. This feature tailors the app to each child's preferences, making the experience more engaging and age appropriate.

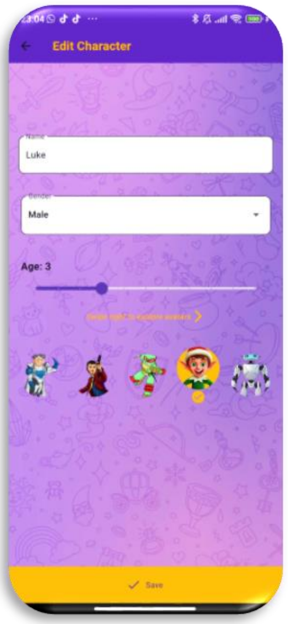


**3. Character Customization:**

Dreamy Tale goes beyond the main character, allowing users to add secondary characters and other main characters to enhance the richness of each bedtime story. This feature provides a dynamic and interactive storytelling experience.

**3.1 Character Modification and Deletion:**

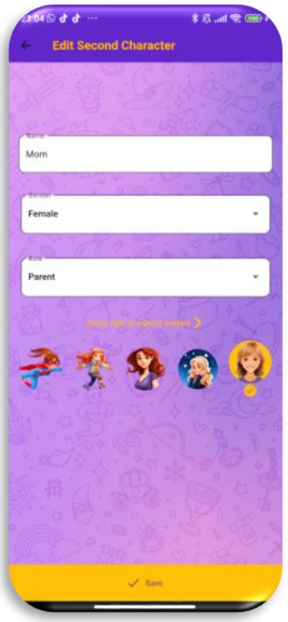
Users can easily modify the details of both main and secondary characters within the app. This includes updating names, appearances, and other characteristics to keep the storytelling experience dynamic.
Additionally, a feature is implemented to allow users to remove characters if they wish to reshape the narrative or streamline the cast for next stories. This ensures flexibility and creativity in crafting personalized bedtime stories.



Name: Luke

Gender: Male

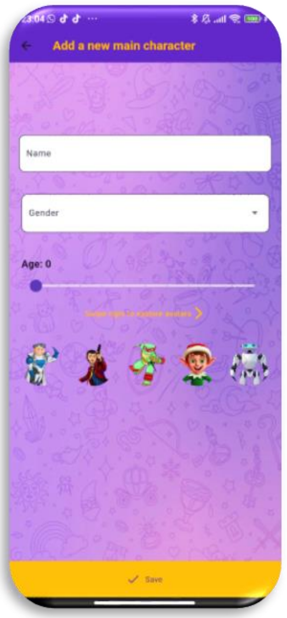
Age: 3



Name: Mom

Gender: Female

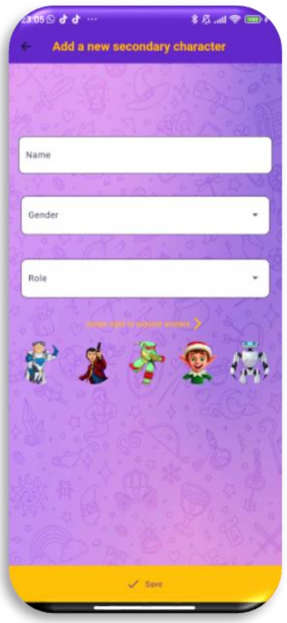
Role: Parent



Name

Gender

Age: 0



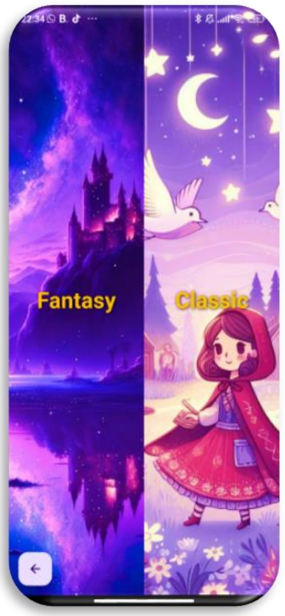
Name

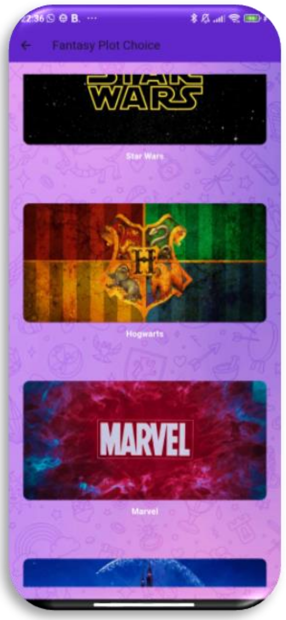
Gender

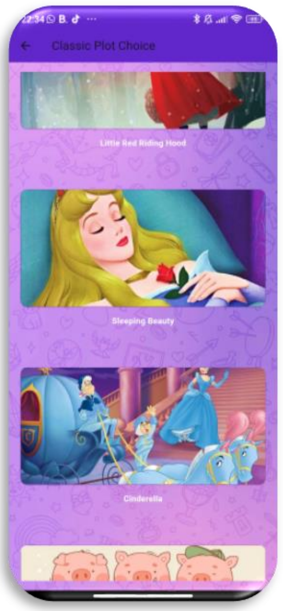
Role

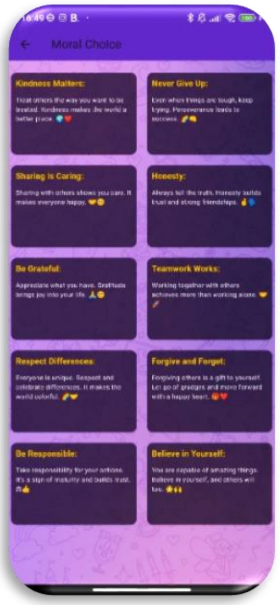
**4. Story Personalization:**

Users have the ability to customize each story by selecting the genre, context, and moral. This feature ensures that every bedtime narrative is uniquely crafted to align with the user's preferences, creating a truly personalized storytelling experience.









Kindness Matters: Treat others the way you want to be treated. Kindness makes the world a better place.

Never Give Up: Even when things are tough, keep trying. Perseverance leads to success.

Sharing is Caring: Sharing with others shows you care. It makes everyone happy.

Honesty: Always tell the truth. Honesty builds trust and strong friendships.

Be Grateful: Appreciate what you have. Gratitude brings joy into your life.

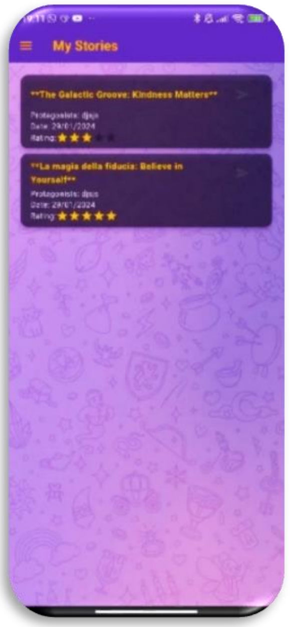
Teamwork Works: Working together with others achieves more than working alone.

Respect Differences: Everyone is unique. Respect and celebrate differences. It makes the world colorful.

Forgive and Forget: Forgiving others is a gift to yourself. Let go of grudges and move forward with a happy heart.

Be Responsible: Take responsibility for your actions. It's a sign of maturity and builds trust.

Believe in Yourself: You are capable of amazing things. Believe in yourself, and others will too.


My Stories

**The Galactic Groove: Kindness Matters**
Protagonists: djsjs
Date: 29/01/2024
Rating: ★★★

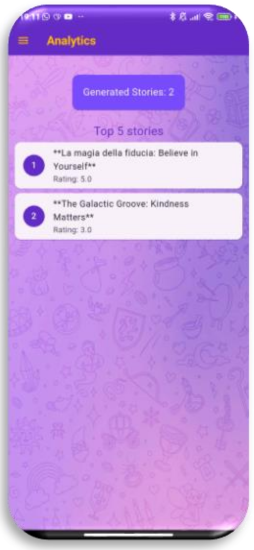
**La magia della fiducia: Believe in Yourself**
Protagonists: djsss
Date: 29/01/2024
Rating: ★★★★★

## 5. Story Archive:

Dreamy Tale provides an archive feature, allowing users to store and revisit their favorite bedtime stories. This creates a cherished library of memories that can be enjoyed repeatedly.

### 5.1 Story Deletion from Archive:

Users have the capability to remove stories from their archive as preferences evolve. This dynamic feature allows for the management of the story library, keeping it curated with the most cherished tales.


Analytics

Generated Stories: 2

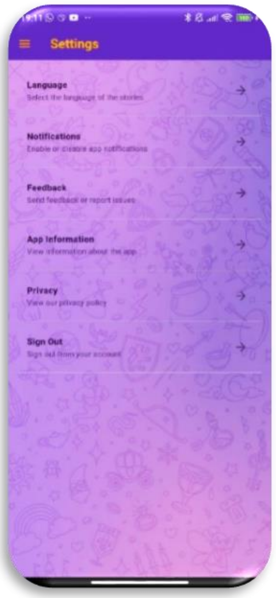
Top 5 stories

1 **La magia della fiducia: Believe in Yourself**
Rating: 5.0

2 **The Galactic Groove: Kindness Matters**
Rating: 3.0

## 6. Story Ratings:

Dreamy Tale empowers users to express their opinions by providing a rating system for each story. This feature allows parents and children to share their feedback, contributing to a community-driven and quality-assured storytelling experience.


Settings

Language
Select the language of the stories

Notifications
Enable or disable app notifications

Feedback
Send feedback or report issues

App Information
View information about the app

Privacy
View our privacy policy

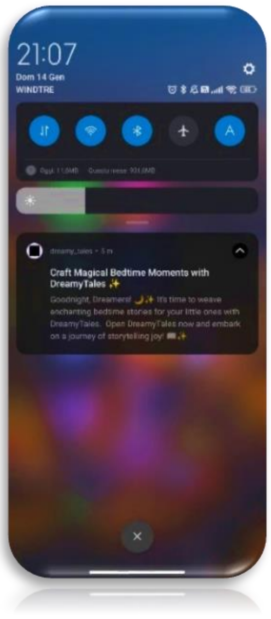
Sign Out
Sign out from your account

## 7. Settings:

Users can personalize their Dreamy Tale experience through a settings menu. This includes options for language preferences, notification settings, and other customizable features, ensuring a tailored and user-centric interface.

## 8. Notifications:

Dreamy Tales keeps users engaged with timely and relevant notifications. Whether it's a reminder for bedtime or updates on new story features, notifications enhance the overall user experience and keep families connected with the app.

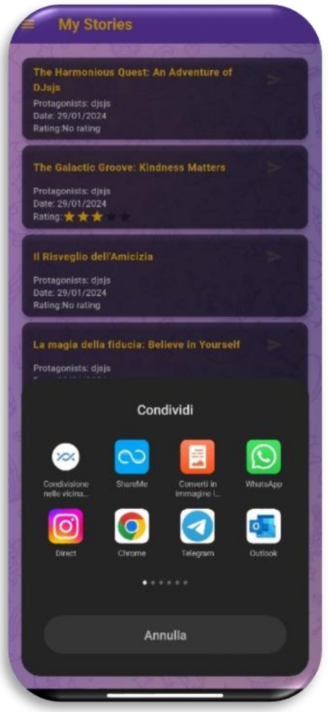


Craft Magical Bedtime Moments with DreamyTales ✨

Goodnight, Dreamers! 🌙✨ It's time to weave enchanting bedtime stories for your little ones with DreamyTales. Open DreamyTales now and embark on a journey of storytelling joy! 📖✨

## 9. Save to Archive:

Users can save their preferred stories to a personal archive for future enjoyment. This feature enables the creation of a curated collection of cherished tales, easily accessible whenever desired.



The Harmonious Quest: An Adventure of DJsjs
Protagonists: djsjs
Date: 29/01/2024
Rating:No rating

The Galactic Groove: Kindness Matters
Protagonists: djsjs
Date: 29/01/2024
Rating: ★★★☆☆

Il Risveglio dell'Amicizia
Protagonists: djsjs
Date: 29/01/2024
Rating:No rating

La magia della fiducia: Believe in Yourself
Protagonists: djsjs

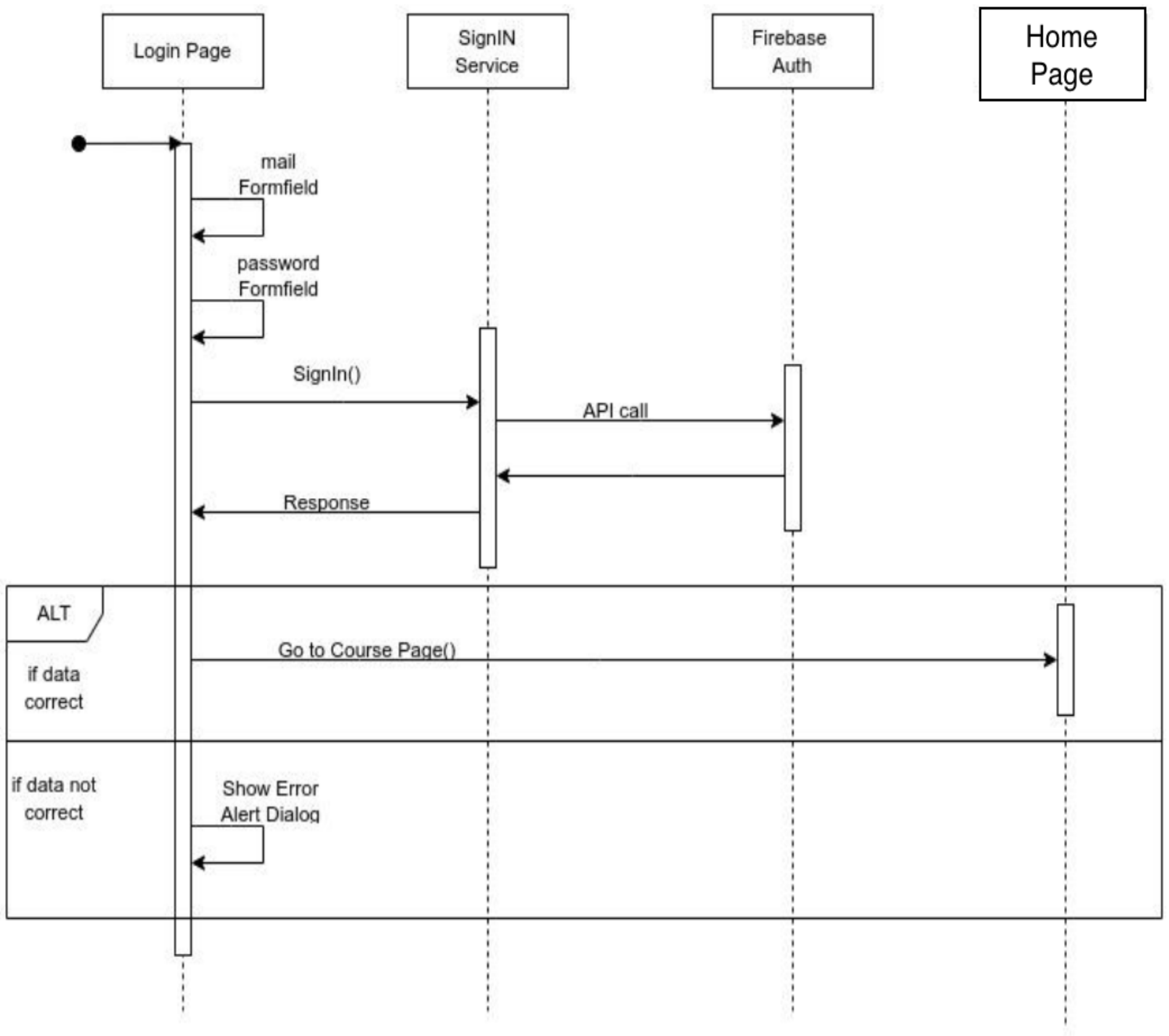
## 10. Social Sharing:

Dreamy Tale facilitates social engagement by allowing users to share their favorite stories seamlessly on popular platforms such as WhatsApp, Facebook, and other social media channels. This feature enhances the collaborative and social aspects of the storytelling experience, enabling users to spread the magic of Dreamy Tale within their networks.

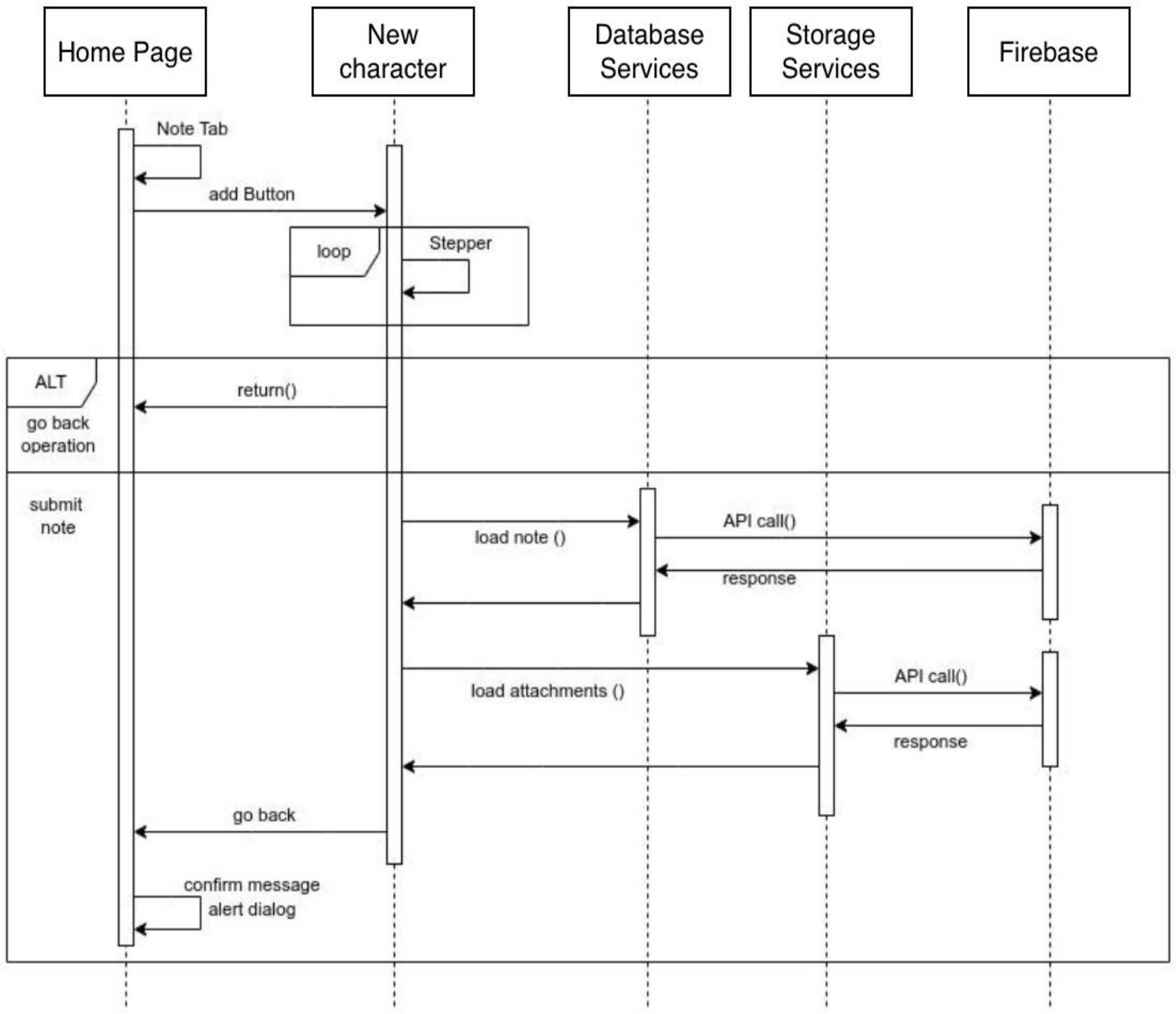
## 2.2 Sequence Diagrams

Sequence diagrams offer a visual representation of how different components or objects interact and cooperate within our system; these diagrams provide a clear understanding of the system's behavior, communication patterns, and the responsibilities of each element. For these reasons, the sequence diagram of the most relevant actions of our application is shown.

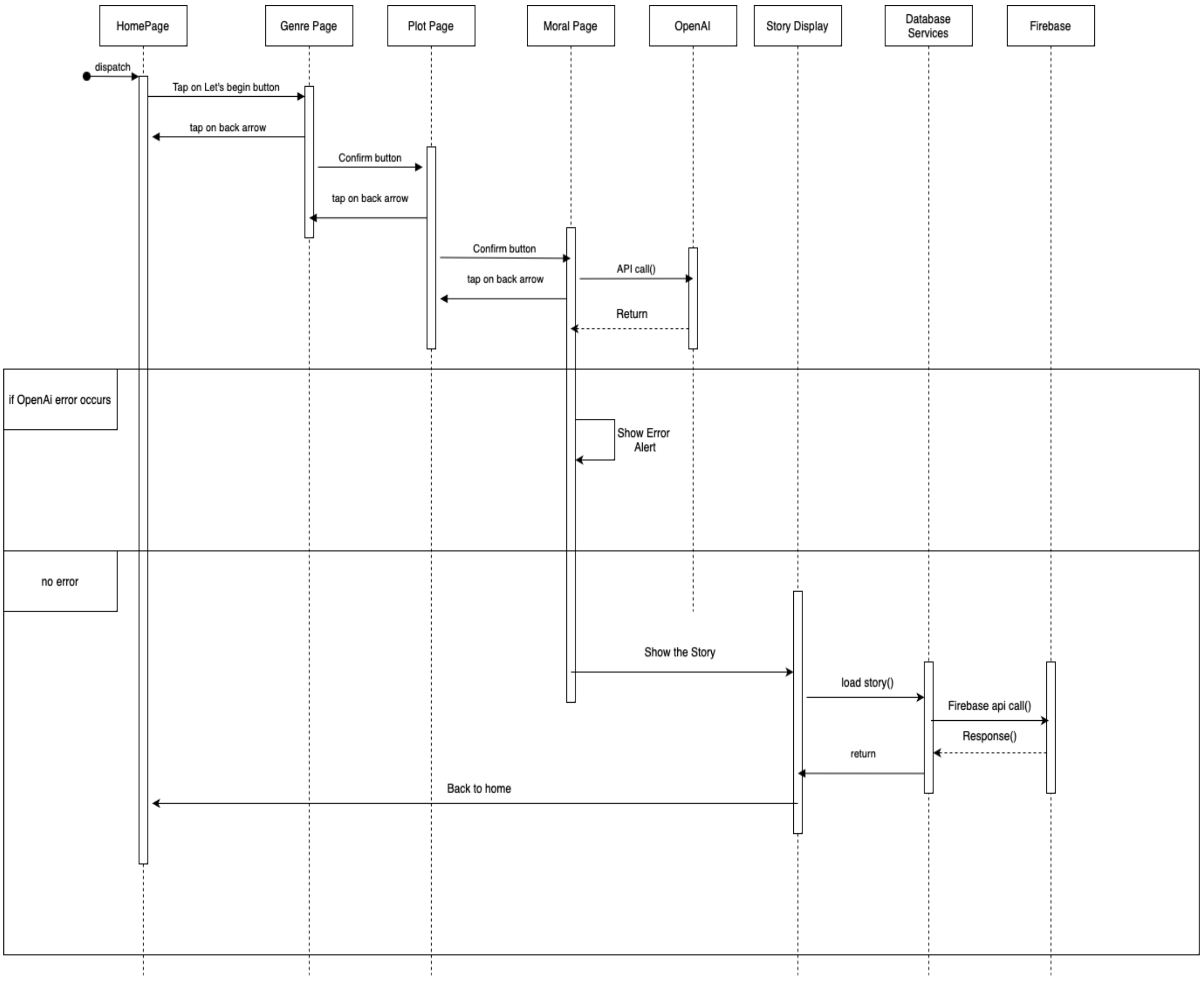
### 2.2.1 Login



### 2.2.2 Add new Character



## 2.2.3 New Story



## 2.3 Requirements

**Domain Requirements (World controlled):**

- **I.** The application should work correctly on different kind of compatible devices (smartphone and tablet)
- **II.** The application should run correctly multi-platform, both IOS or Android
- **III.** User should have a compatible device where run the application.
- **IV.** The user should be able to see data from the device in real time, and to store them in a database.
- **V.** Device should have a correct-working internet connection which allow communications with Cloud Storage

**Software Requirements (Machine controlled):**

- **I.** The application must work only after the authentication of a user, to keep track of all the changes.
- **II.** System could establish a connection with Cloud Storage
- **III.** In case of error in receiving the data from Cloud storage, software warns user through an error message.
- **IV.** The application supports multi-platform, compatible both IOS or Android

**Authentication Requirements:**

- **V.** Once the app is in the login/register page, a login form must be shown to allow the user's authentication.
- **VI.** A link to the signup form must be present, to be used by a not yet registered user.
- **VII.** If the credentials are wrong, an alert message should be displayed.
- **VIII.** The input field for the password must have the "hidden" font, but if the user wants to, it can be displayed clearly by pressing an icon.
- **IX.** Once a user is logged, the list of the character he has created must be shown in the home page (divided in main and second character)

**Story Requirements:**

X. The system allows to create a new personalized story by choosing the plot, genre, and a moral lesson.

XI. The system allows user to retrieve all previous story generated.

XII. The system allows to listen the story by a voiceover system.

XIII. The system allows to delete one story by long pressing on its preview and confirming the deletion.

**Character Profiling Requirements:**

XIV. The system allows to add more character by pressing the appropriate button.

XV. The system allows to update a character by pressing on his avatar.

XVI. The system allows to delete one character by long pressing on its avatar and confirming the deletion.

**World Phenomena (User controlled):**

I. User could sign In / Register into the app.
II. After the registration process, the user could create his first main character, selecting his name, gender, age and an Avatar.
III. The user decides how many characters add into his profile.
IV. The user decides how to personalize his story.
V. User can navigate through different pages of app through taps and swipe on device's screen.

**Non-Functional Requirements**

- **Portability**: the app must be used on both Android and iOS smartphones.
- **Extensibility**: it is possible to use any kind of device for measuring data, both physical and virtual. The only requirement is that of sending data through Wi-Fi with a specific format, that will be mentioned after in this document.

- **Maintainability**: the code should be easily readable.
- **Reliability**: the system must be safe and not broken by anyone.
- **Efficiency**: the app should use the least resources possible.

# 3A Architectural Design

## 3.1 Client server Paradigm

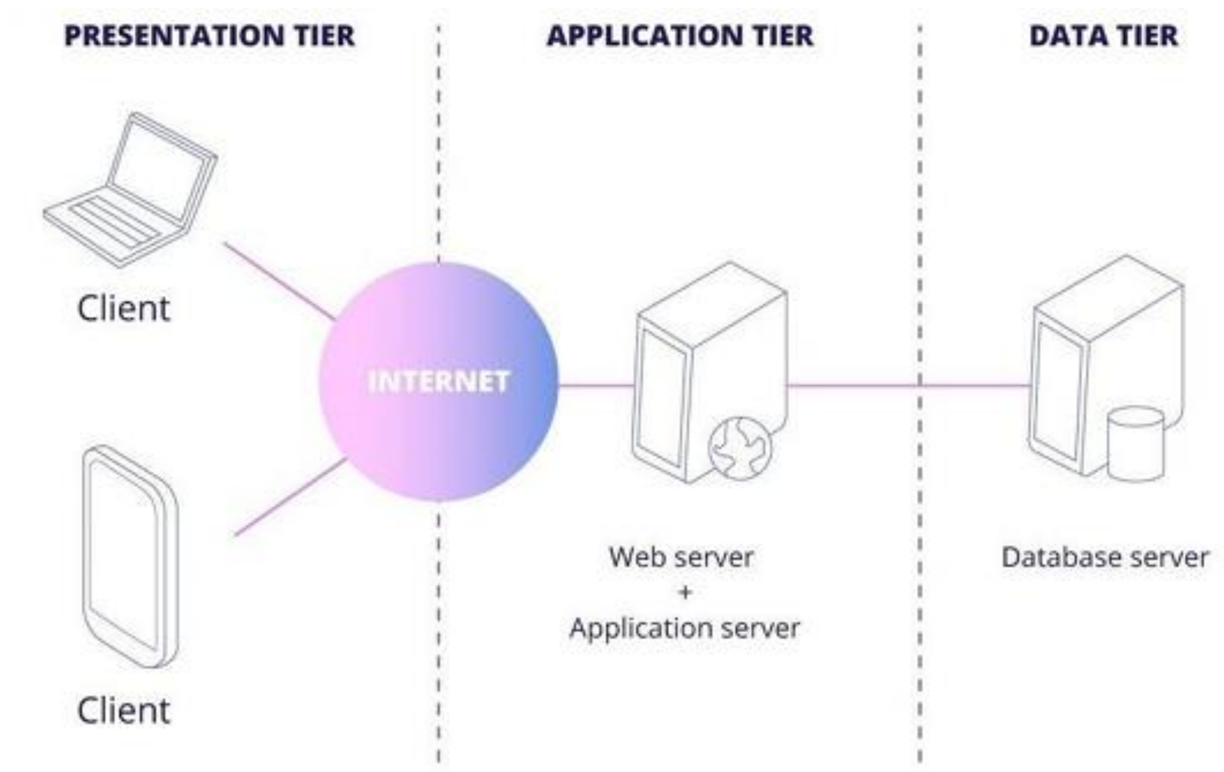


*Figure 1: Client-Server Paradigm*

The system is a distributed application which follows the commonly known client-server paradigm. We have a single type of client, a thin mobile application, which permits exploiting all the functionalities of the system. Instead, the remote server is fat and contains all the data management and business logic.

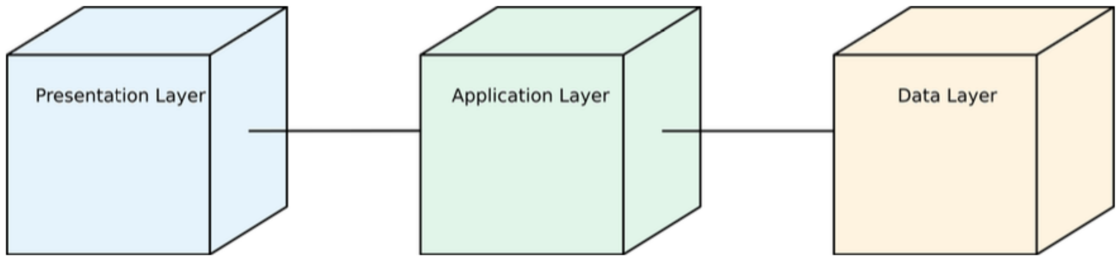


Figure 2: Three layers application

In figure 2 the S2B layers are shown. Going more into details:

- **Presentation Layer**: it manages the presentation logic and, consequently, all the interactions with the end user. It is also called rendering layer.
- **Application Layer**: it manages the business functions that the system must provide. It is also called Logic Layer.
- **Data Layer:** it manages the storage and relative access to data.

These three layers are physically separated through the installation of them on different tiers. A tier is a physical (or a set of) machine, with its own computational power.

The application described in this document is composed of three tiers, which are shown in figure 3.

Application servers the application tier, also known as the logic tier or middle tier, is where information collected in the presentation tier is processed (sometimes against other information in the data tier) using business logic, which means a specific set of business rules. The application tier can also add, delete or modify data in the data tier, in fact, this tier is able to communicate with the data tier by using API calls.

The core of the software is installed on the user's device, which is represented in the Presentation Layer. It connects to the provided API (passing through a Firewall) installed in several replicated application servers in the Application Layer. The request is handled by a Load Balancer, which routes the request to the correct server.

Finally, the application servers are connected to a cluster of distributed databases on the last layer, which is the Data Layer, through the DBMS APIs.

The application servers are expected to be stateless, according to the REST standard definitions. For accessing the data, they will use an ORM programming technique, exploiting the advantages of the object-oriented paradigm.

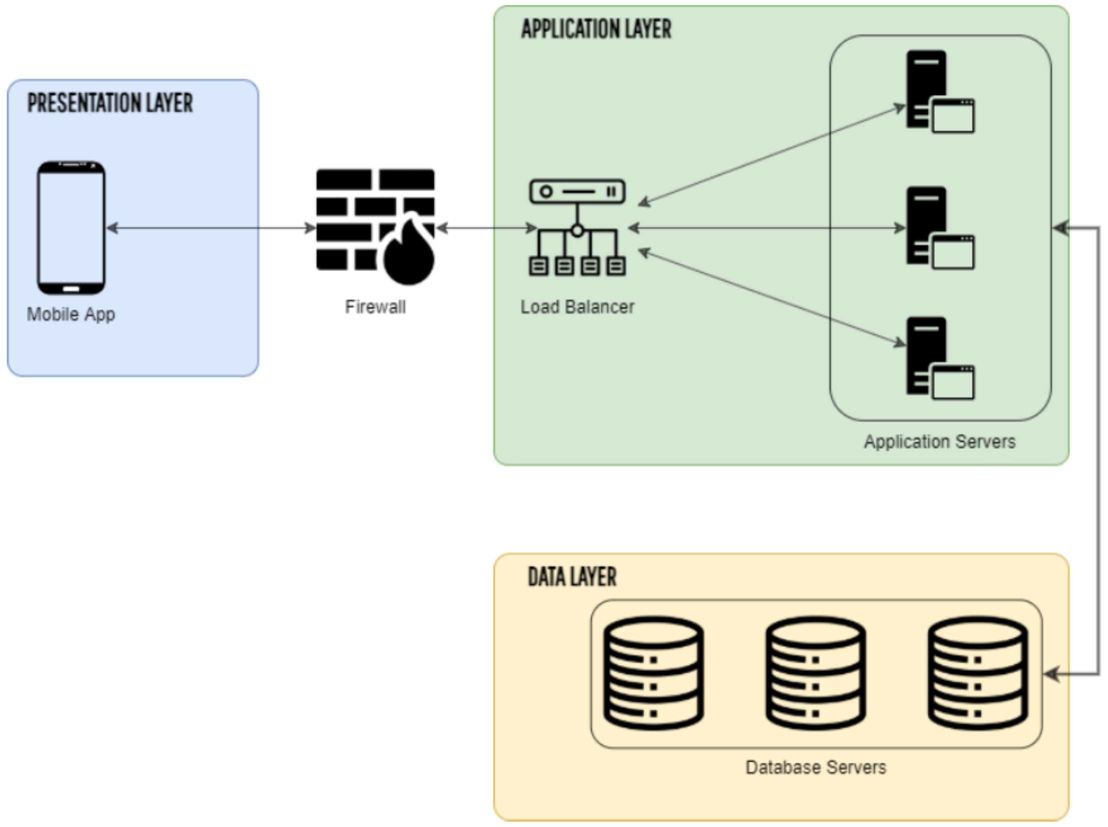


Figure 3: Architecture of the application

## 3.2 Architectural Style and Patterns

Three-tiered architecture

We chose this architecture for many reasons:

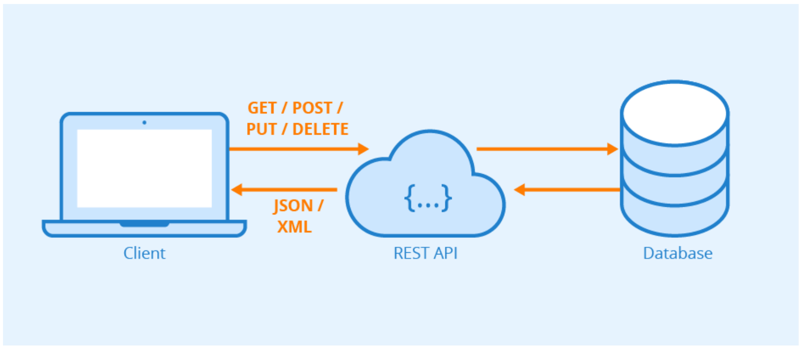
- **Flexibility**: once the interfaces of the S2B are defined, then the interior logic is dependent from outside. This fact implicates that each module can be improved without changing all the others.
- **Scalability**: an application divided on several tiers guarantees that the approach of scaling the architecture is adopted only for the most critical components. The result obtained maximizes the performance but also minimizes the costs.
- **Load Distribution**: the presence of several application servers, pre-ceded by a load balancer, guarantees an acceptable division of requests. Otherwise, the presence of a single node means that node can become over-requested, sending the entire system down.

## 3.3 RESTful Architecture

The RESTful architecture is based on the stateless principle, in which the server does not contain any information about the state of the client, that is managed directly on the client side.

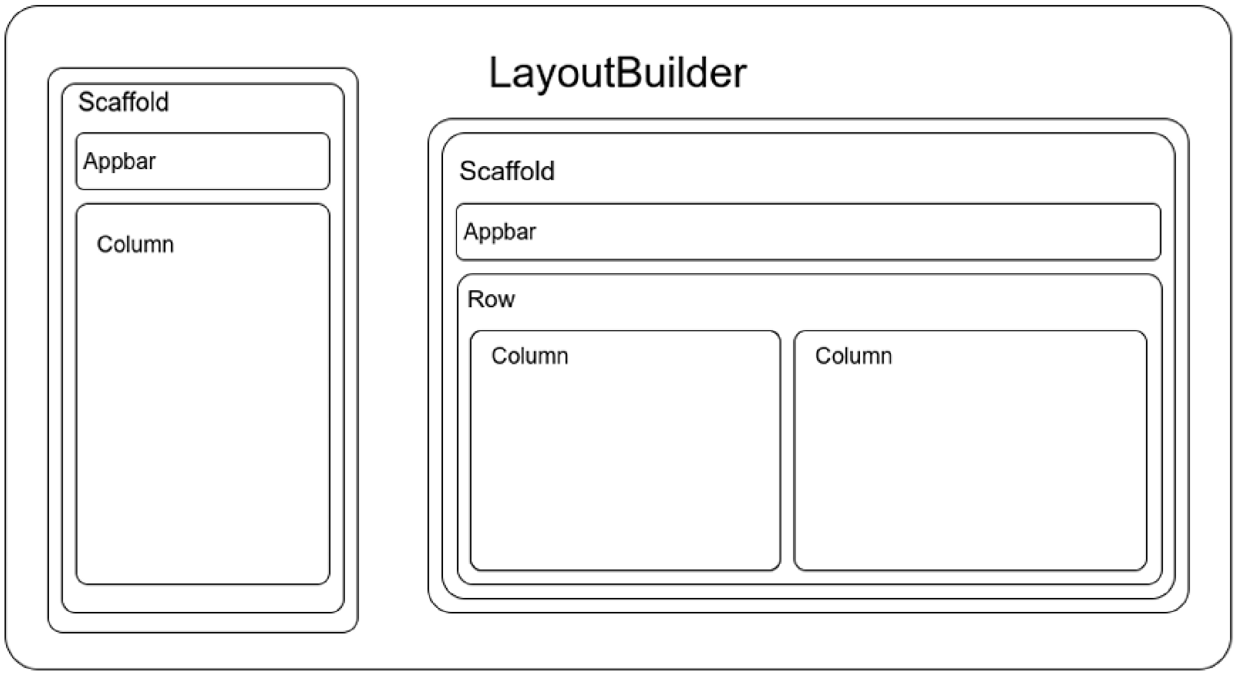
This behavior guarantees less computational load on the server and a dynamic attitude of the services.

The application is then intended to be developed through client-side programming, which means that all requests and updates of the page are made on client side.



## 3.4 Widget Architecture

In this section the general widget organization of the pages is shown. The use of this widget framework is used to better generalize on different kinds of screen size and help with code structure for better reuse of it.



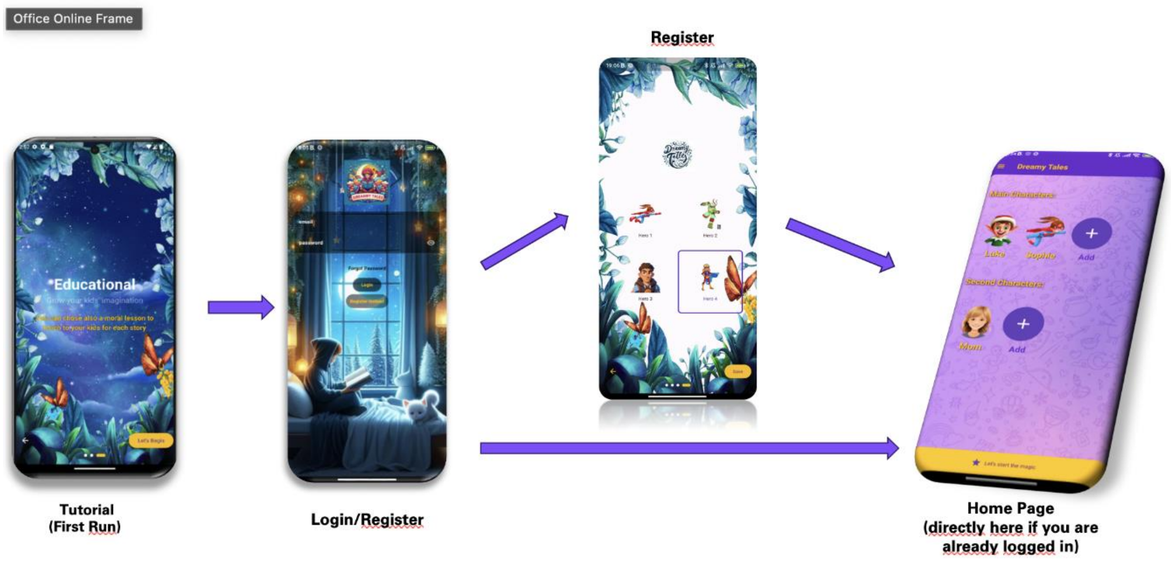
As can be seen in the User Interface Paragraph (next one), most of the tablet screens are composed of different smartphone pages positioned side by side. To implement this design, we have chosen the above widget framework: a scaffold with an app bar that remains constant in most of the pages, and a column that includes all the main widgets used in the page; this column is often wrapped within a general widget class to be reused in the tablet implementation. Both smartphone and tablet pages are then wrapped in a LayoutBuilder widget that allows us to select the right design dynamically according to the screen size and in a transparent way with respect to the design of the page. The choice of wrapping the whole page and not only the different widgets was made in order to achieve an optimal trade-off between reuse of the code and design flexibility.

# 3B User Interface Design

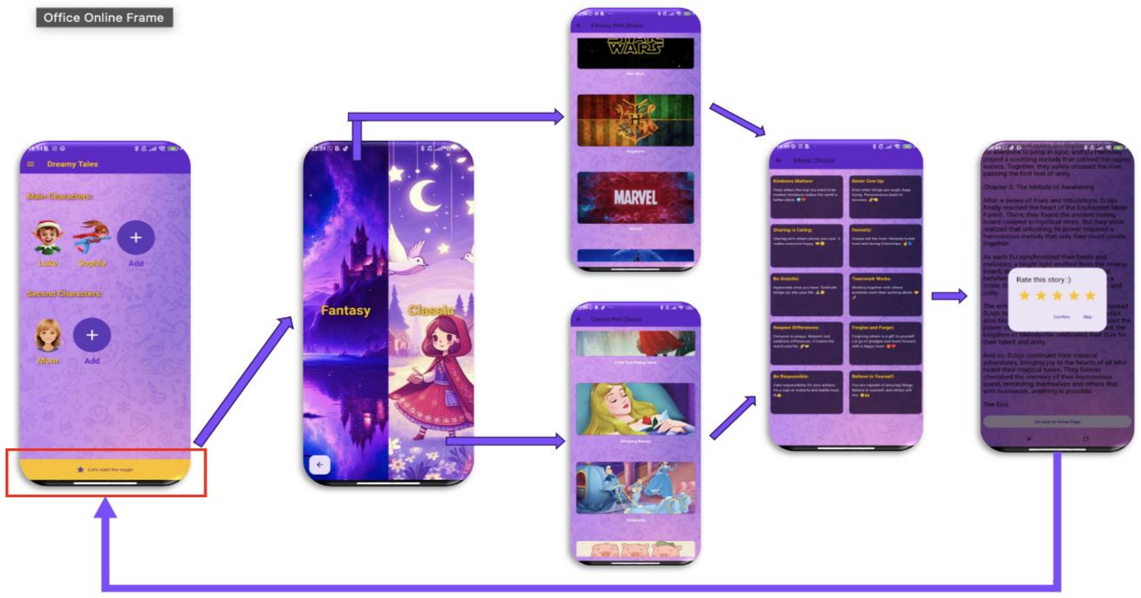
## 3.1 Introduction

The aim of this section is to show the design of the main screens of the user application, describing the **flow** of the main functionalities for which it is intended. The flow is created according to specific and illustrated input of the end user.

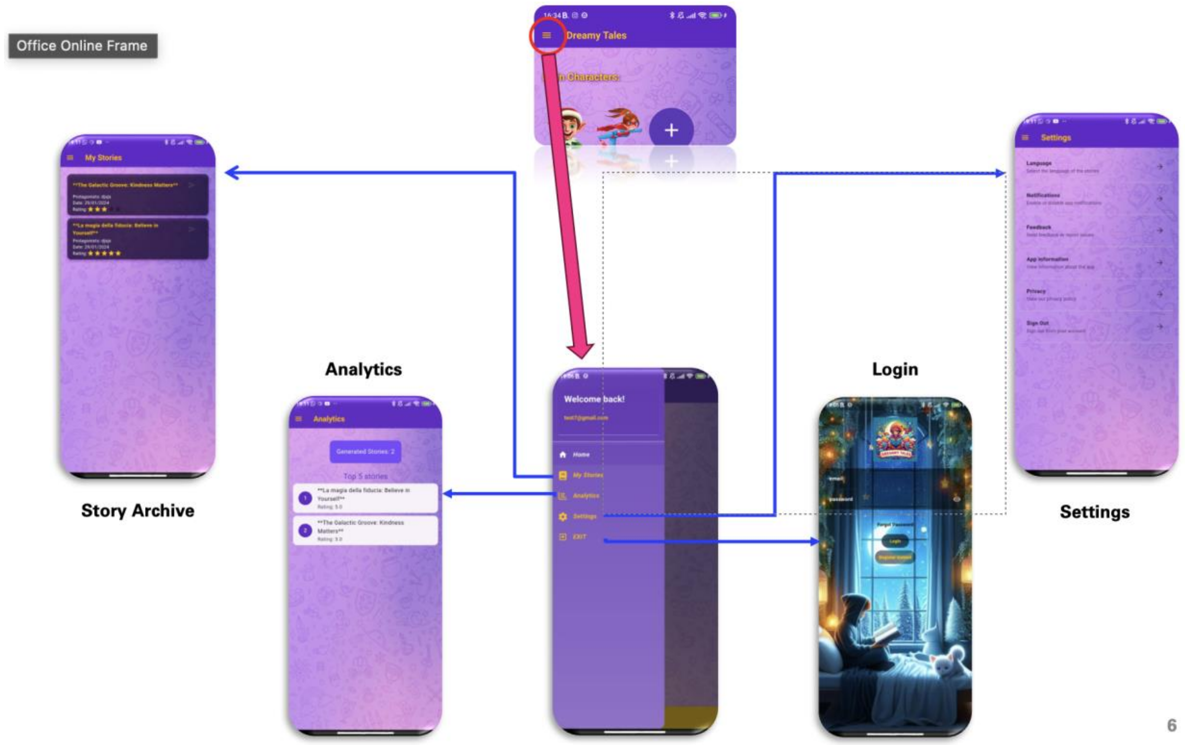
### 3.2 Tutorial, Login and Profiling Phase



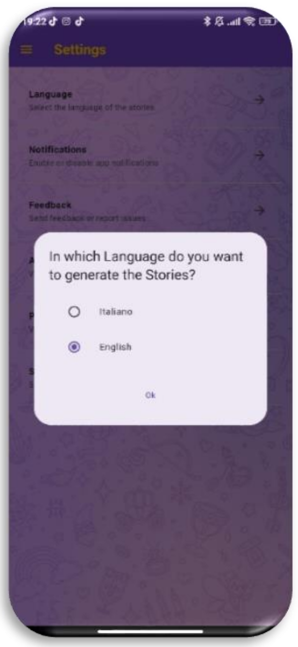
### 3.3 Generation of a new story



## 3.4 Insert, Update or Deletion of a (Main / second) character

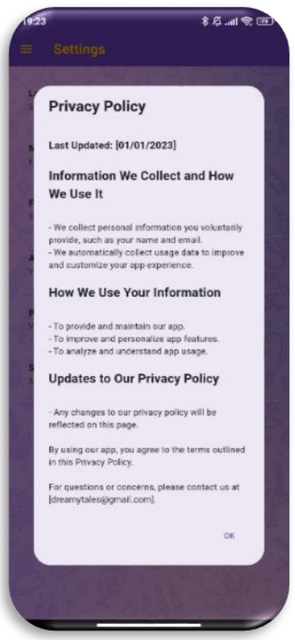


## 3.5 Settings Page


Settings
Language
Select the language of the stories
Notifications
Enable or disable app notifications
Feedback
Send feedback or report issues

In which Language do you want to generate the Stories?

Italiano [✓ English]



**Privacy Policy**

**Last Updated: [01/01/2023]**

**Information We Collect and How We Use It**

- We collect personal information you voluntarily provide, such as your name and email.
- We automatically collect usage data to improve and customize your app experience.

**How We Use Your Information**

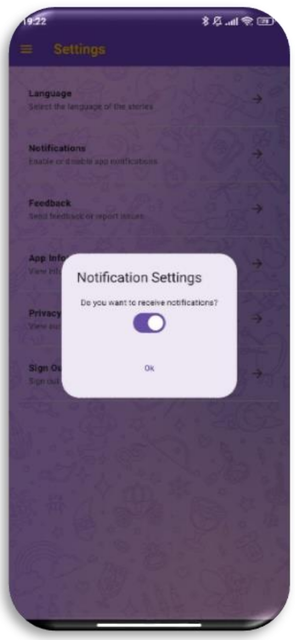
- To provide and maintain our app.
- To improve and personalize app features.
- To analyze and understand app usage.

**Updates to Our Privacy Policy**

- Any changes to our privacy policy will be reflected on this page.

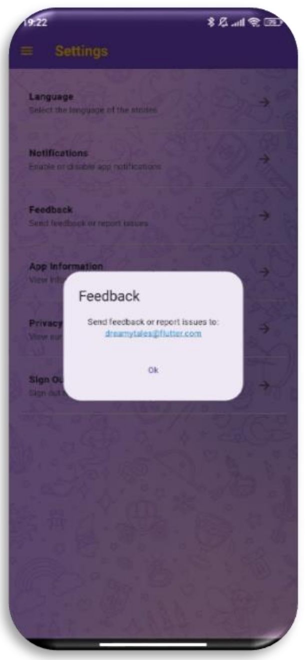
By using our app, you agree to the terms outlined in this Privacy Policy.

For questions or concerns, please contact us at [dreamytales@gmail.com].


Settings
Language
Select the language of the stories
Notifications
Enable or disable app notifications
Feedback
Send feedback or report issues

Notification Settings

Do you want to receive notifications?


Settings
Language
Select the language of the stories
Notifications
Enable or disable app notifications
Feedback
Send feedback or report issues
App Information

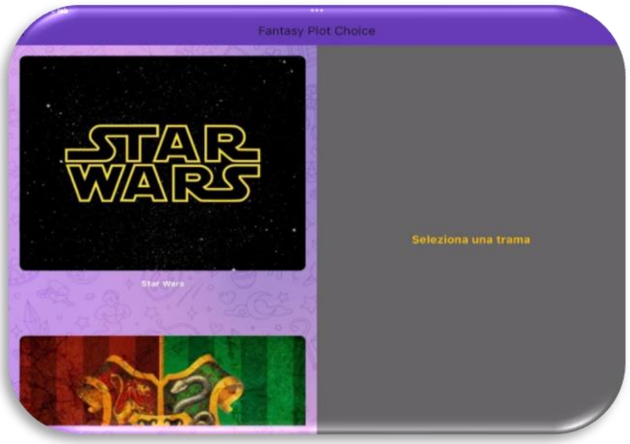
Feedback

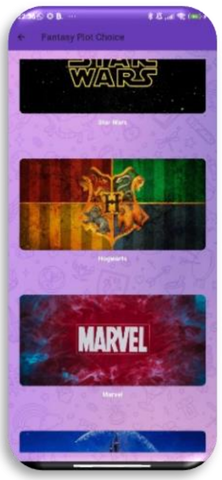
Send feedback or report issues to: dreamytales@flutter.com

## 3.6 Adaptive Layout

For some pages, we have adopted an adaptive approach leveraging the 'flutter_device_type' framework to identify the type of device with which the user is interacting, specifically addressing smartphones and tablets. Below are some images illustrating the differences in page layout based on the device.

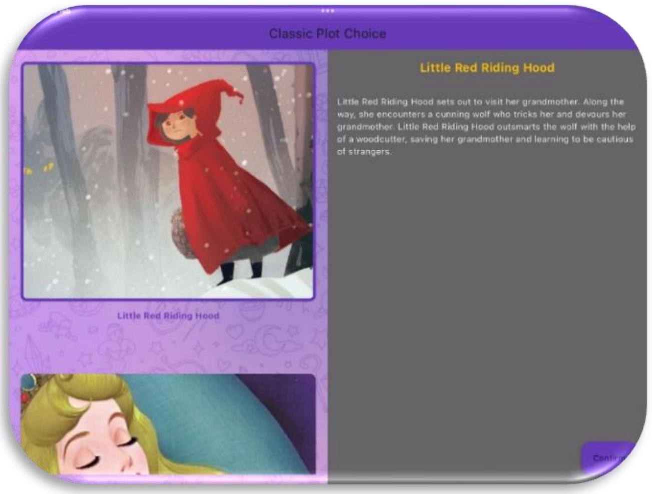
**Fantasy Plot Page**



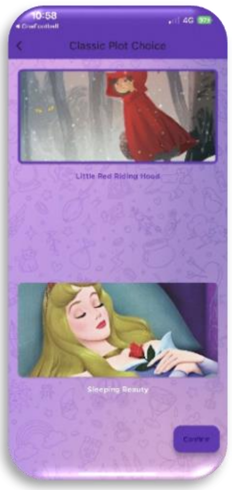


*Tablet Version* *Smartphone Version*

**Classic Plot Page**

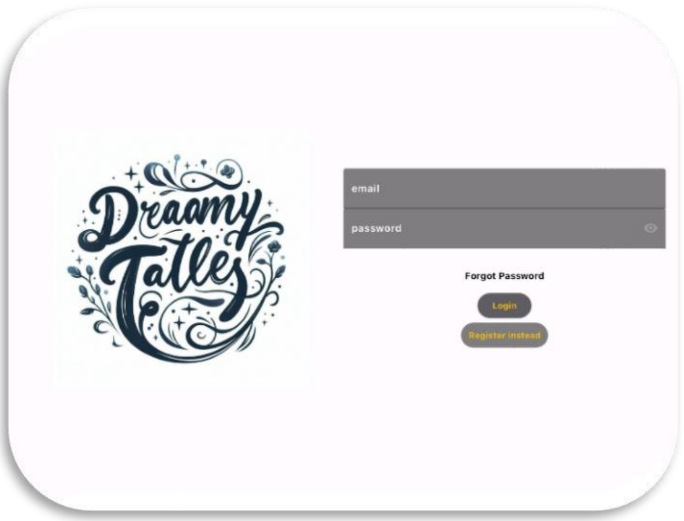


Little Red Riding Hood sets out to visit her grandmother. Along the way, she encounters a cunning wolf who tricks her and devours her grandmother. Little Red Riding Hood outsmarts the wolf with the help of a woodcutter, saving her grandmother and learning to be cautious of strangers.

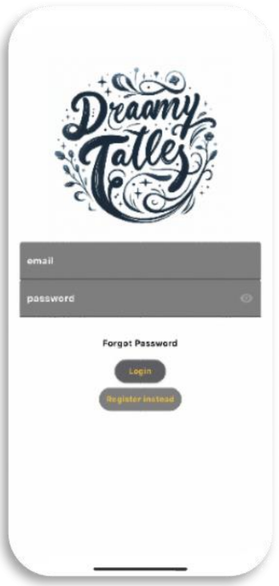


*Tablet Version* *Smartphone Version*

### Login Page



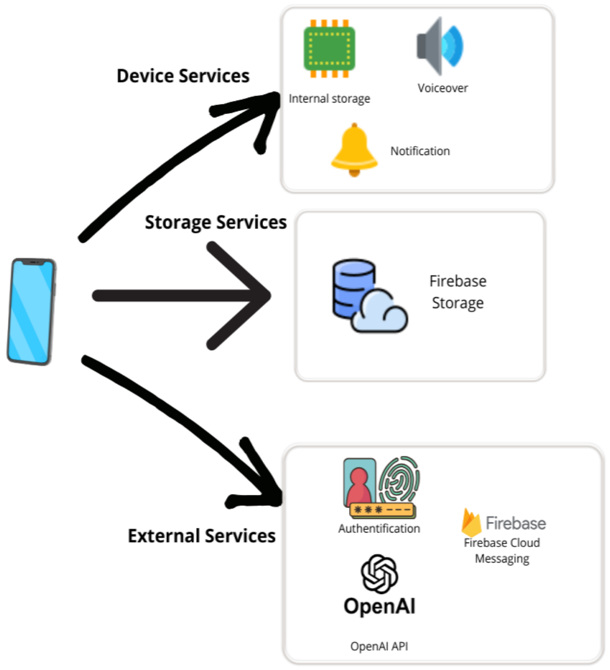
*Tablet Version*



*Smartphone Version*

For many other pages we adopt a responsive approach to adapt the interface to each device using flutter function Mediaquery.

# 4 Application services



## 4.1 Internal Organization

The most important portion of data is stored on the server, therefore the needed data are obtained through fetch call and stored temporarily in local and global variables just to be shown in the related views.

## 4.2 External Database Services

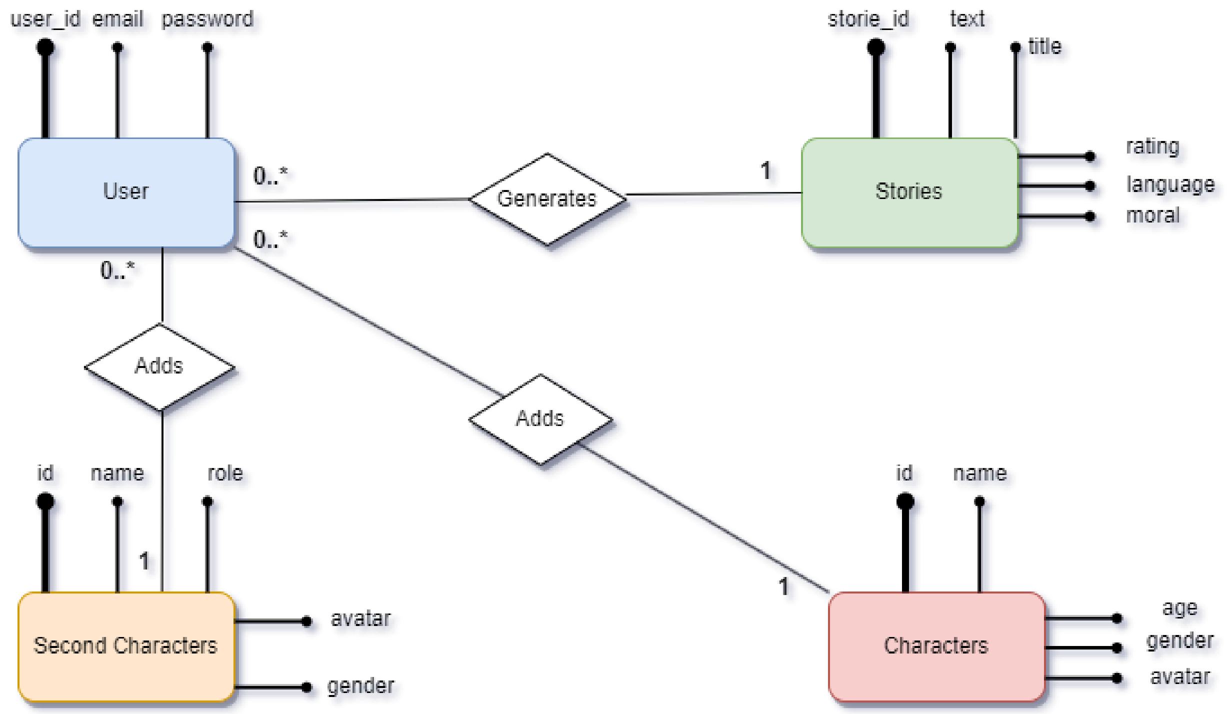
Dreamy Tales relies on Firebase as its external database service, ensuring secure and efficient storage of character and story data. All character profiles and stories are systematically organized within Firebase's server, each associated with a unique user ID for seamless retrieval during fetch operations.

User-specific data, such as character details and personalized stories, is linked to the user's ID, guaranteeing a personalized experience post-login. This association facilitates quick and precise data access, providing users with their tailored content upon authentication.

The connection to the Firebase server is established using Firebase's robust and secure APIs. Dreamy Tales utilizes Firebase Authentication to securely identify and authenticate users, enabling a seamless and protected connection to the database server. This approach ensures that all user-related data, including characters and stories, is efficiently managed and retrieved, contributing to a fluid and personalized storytelling experience within the Dreamy Tales app.

### 4.2.1 Cloud Firestore

Cloud Firestore is a document-based database that stores all the relevant data displayed in the application. In particular, the internal structure of the collections is the following:



**User**: this collection stores information about registered users, i.e. email, userid and the crypted password (the authentication process is entrusted to firebase.auth)

**Stories:** this collection stores the stories created by the users, and all their information, i.e, storie_id, text, title, rating (if rated), story language, and the moral. The collection contains also the user_id of the user who generated the story (this field is useful to retrieve the list of stories created by a given user)

**Character:** this collection contains the information about the character created by a user, i.e. id, name, age, gender, avatar (the file path)

**Second Character:** this collection contains the information about the character created by a user, i.e. id, name, age, gender, avatar (the file path), and the role

Both main and second character have a field user_id associated to the user who created those characters, in this way is easier to retrieve the characters related to a user.

## 4.3 Shared Preferences

Shared preferences is a plugin that provides persistent storage for simple data represented in key-value format in device's memory.
In this application it has been useful for storing users' preferences:

- choice of genre of the story to generate
- choice of the language in which the user wants to generate the story
- choice of plot of the story
- choice of the moral lesson

When a user updates one of these options in the dedicated page, a Boolean value is assigned to the specific key and stored in memory so that, when the user closes and reopens the application, the chosen settings are always restored.

## 4.4 External Services

### 4.4.1. Firebase Authentication

Firebase Authentication provides backend services to authenticate users to our app. It has been used to support authentication using email and password which are then stored when a user completes the registration phase, together with a unique generated id.

### 4.4.2 Firebase Cloud Messaging (FCM) for Notifications:

Dreamy Tale leverages Firebase Cloud Messaging (FCM) to seamlessly manage and deliver notifications, enhancing user engagement and interaction. FCM provides a reliable and scalable solution for sending push notifications to both Android and iOS devices, ensuring that users stay connected with the app's latest updates.

### 4.4.3 OpenAI API for the story generation

The integration of the OpenAI API plays a pivotal role in the story generation process within our project. This artificial intelligence API allows our system to enrich narratives with original and engaging content. Leveraging advanced natural language processing algorithms, the OpenAI API tailors stories to user preferences, providing a unique and personalized reading experience. The incorporation of this powerful API enables our project to transcend the constraints of pre-defined narratives, offering a dynamic and immersive storytelling experience, thereby enhancing the storytelling journey within the Dreamy Tales app.

### 4.5 Dependencies

The most significant flutter packages included in the application are listed below:

| | |
|---|---|
| firebase_core | Used to initialize Firebase services |
| Firebase_auth | Used for authentication with email and password |
| flutter_test integration_test | Used to perform widget tests Used to perform integration tests |
| firebase_messaging | Necessary to use the Firebase Cloud Messaging API |
| responsive_framework | Used to adapt the screens in a responsive way |
| share | Used to give the users the possibility to share to their stories on the social |
| cloud_firestore | Used to initialize the firestore |
| mockito | For the authentication test |
| shared_preferences | To keep same user preferences used to generate personalized stories |
| http | Used to create an http connection |
| flutter_rating_bar | To generate the rating bar of the rated stories |
| flutter_tts | Used for the voiceover of the stories |
| dart_openai | Used to generate the connection between the app and API of openAI |

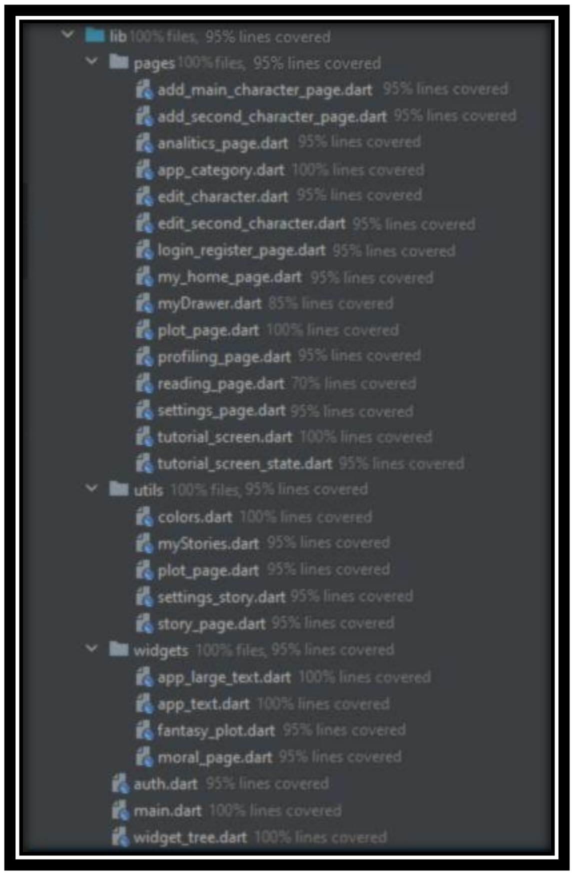
# 5 Testing Campaign

Three types of tests will be performed:

- **Unit testing**: it ensures that the end-user (customers) can achieve the goals set in the business requirements, which determines whether the software is acceptable for delivery or not.
- **Widget testing**: the goal of a widget test is to verify that the widget's UI looks and interacts as expected. Testing a widget involves multiple classes and requires a test environment that provides the appropriate widget lifecycle context.
- **Integration testing**: it ensures that an entire, integrated system meets a set of requirements. It is performed in an integrated hardware and software environment to ensure that the entire system functions properly.

## 5.1 Unit testing

We have tested every component of the app with Unit Testing reaching a coverage around 90/95%



```
lib 100% files, 95% lines covered
  pages 100% files, 95% lines covered
    add_main_character_page.dart 95% lines covered
    add_second_character_page.dart 95% lines covered
    analitics_page.dart 95% lines covered
    app_category.dart 100% lines covered
    edit_character.dart 95% lines covered
    edit_second_character.dart 95% lines covered
    login_register_page.dart 95% lines covered
    my_home_page.dart 95% lines covered
    myDrawer.dart 85% lines covered
    plot_page.dart 100% lines covered
    profiling_page.dart 95% lines covered
    reading_page.dart 70% lines covered
    settings_page.dart 95% lines covered
    tutorial_screen.dart 100% lines covered
    tutorial_screen_state.dart 95% lines covered
  utils 100% files, 95% lines covered
    colors.dart 100% lines covered
    myStories.dart 95% lines covered
    plot_page.dart 95% lines covered
    settings_story.dart 95% lines covered
    story_page.dart 95% lines covered
  widgets 100% files, 95% lines covered
    app_large_text.dart 100% lines covered
    app_text.dart 100% lines covered
    fantasy_plot.dart 95% lines covered
    moral_page.dart 95% lines covered
  auth.dart 95% lines covered
  main.dart 100% lines covered
  widget_tree.dart 100% lines covered
```

Figure xx. Coverage of Unit Testing

## 5.2 Integration testing

We also implemented a couple of integration tests that help to identify regression bug when a new feature is introduced and the test some key functionality of our application.

- **Authentication Process**
  - Signup: We use the library Mocktail to mock an instance of FirebaseApp, and to complete a signup process successfully ending to the profiling page
  - Login: We use the library Mocktail to mock an instance of FirebaseApp, and to complete a login process successfully ending to the Home page directly
  - Total Exception tested:
    - *case 'too-many-request':*
    - *case 'user-not-found':*
    - *case 'weak-password':*
    - *case 'email-already-in-use':*
    - *case 'invalid-email':*
    - *case ‘invalid-login-credential:*
    - *case ‘unknown error’*

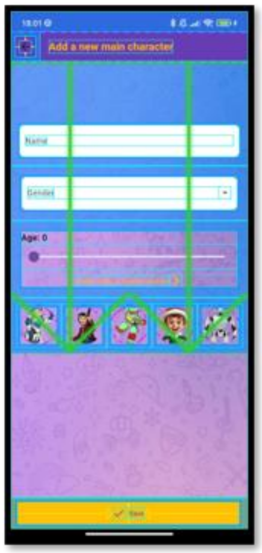
- **Creation of a Story**
  - That tests the entire story generation process, the be sure that the navigation from choosing the genre to calling OpenAI API to generate the story is solid.

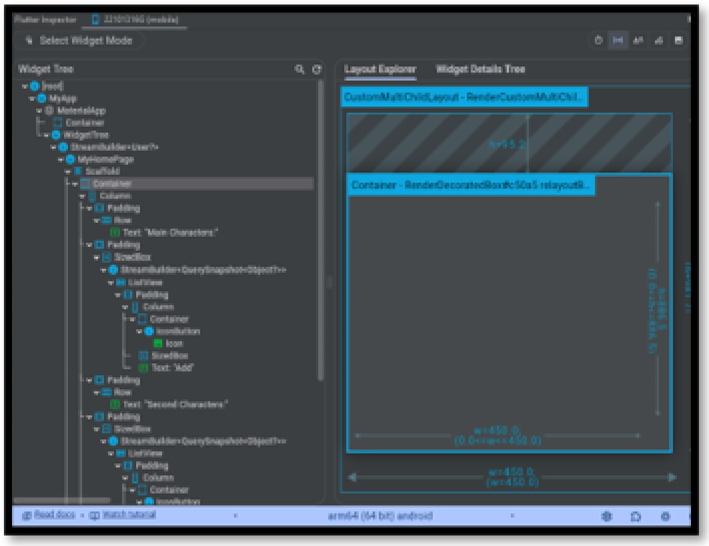
- **Interaction between Views**
  - Navigation through different views of the application to guarantee the correct behavior of button and links (for example Drawer navigation, backward/forward through Homepage, MyStories, Analytics, Setting, LogOut)

- **Retrieve Data**
    - Query test (tested on mockup):
        - *Retrieve correct Main Character*
        - *Retrieve correct Secondary Character*
        - *Retrieve correct User Details*
        - *Retrieve correct stories archive for each user*
        - *Delete Story*
        - *Delete Main Character*
        - *Delete Secondary Character*
        - *Edit Main Character*
        - *Edit Secondary Character*

## 5.3 Other types of testing

- **Flutter DevTools:**
    - Performance: Find peak of low frame due some GUI compilation
    - CPU profiler: sample critical actions and evaluate if it is needed some optimization.
- **Flutter Inspector:**
    - Analyze Widget Tree
    - Layout Explorer to fix positional issues.


Add a new main character
Name
Gender
Age: 0
Save


Flutter Inspector
Select Widget Mode
Widget Tree
Layout Explorer
Widget Details Tree
[root]
MyApp
MaterialApp
Container
WidgetTree
MyHomePage
Scaffold
Container
Column
Padding
Row
Text: "Main Characters:"
SizedBox
ListView
IconButton
Icon
Text: "Add"
Text: "Second Characters:"
Container - RenderDecoratedBox
Read docs
Watch tutorial
arm64 (64 bit) android